**ASSIGNMENT 02 – CREDIT RISK PREDICTION**

Username on miner: shirdisai

Name: Varshaa Shree Bhuvanendar

Rank: 15(At the time of making report)

FI-Score: 0.68

AIM: To perform credit risk prediction using different prediction algorithms

Introduction:

I have used Gradient Boosting algorithm to perform credit risk prediction as the data set has imbalanced data.

Steps Followed:

1. Performed one hot encoding on data
2. covariance of every column with the destination column “credit”
3. Columns with values close to 0 covariance I dropped
4. Since data is imbalanced performed SMOTE algorithm over the dataset
5. I then tried Random Forest, Linear Support Vector Machine, Perceptron and Gradient Boosting
6. Found F1-Score

How did you deal with different features?

Since the dataset is a mix of textual and number-based data. To handle textual data, I have tried both label encoding and one hot encoding.

I found one hot encoding gives better results.

Label encoding: If there are n unique groups in a column of dataset label encoding, assigns values from 0 to n-1 based on alphabetical order. By doing this we are indirectly assigning rank to certain groups.

To avoid this, we use one hot encoding. It simply creates additional features based on the number of unique values in the categorical feature. Each category is represented as a one-hot vector.

In the current dataset I applied it to columns F10 and F11.

Did you exclude any specific features?

To find out which features I can remove I found correlation between credit determinant column and all other columns.

Anything with variance close to 0 like id,F8 and F9 are removed

Table

Description automatically generated

How did you perform model selection and which classifier stood out? Any theoretical reasoning why?

I implemented KNN, Random Forest, Perceptron, Naïve Bayes, Gradient Boosting.

Gradient Boosting stood out.

Reason:

Gradient Boosting works on the wrong prediction. So, if we have large set of biased data. So, in first iteration when the algorithm predicts some of the rows wrong then in next iteration tries to find why it predicted wrong and works on the wrong prediction.

Was there a certain way you dealt with imbalance in the class distributions?

Tried using oversampling.

Oversampling can be defined as adding more copies to the minority class. Oversampling can be a good choice when you don’t have a ton of data to work with.

I tried the SMOTE Synthetic Minority Oversampling Technique.

SMOTE (Synthetic Minority Oversampling Technique) works by randomly picking a point from the minority class and computing the k-nearest neighbors for this point. The synthetic points are added between the chosen point and its neighbors.

Used following library imblearn.over\_sampling

Conclusion:

Gradient Boosting technique can be used as a prediction model for an imbalanced dataset.

For feature selection we use correlation and determine which features to remove.

Also oversampling and undersampling can be applied on the dataset to avoid biasing.

OUTPUT (f1 score):

Text

Description automatically generated

Output Explanation:

* By running the first cell it uses gradient boosting and one hot encoding output from first cell is final output.
* By running second cell the label encoder based output with gradient boosting is obtained.
* Third cell shows correlation over the dataset
* The further cells show accuracy and model fitting on the train\_data for SVM,Perceptron, Random Forest, KNN, Logistic Regression and more.